Analyzing Circuit-aware Microarchitectural Reliability

Taniya Siddiqua, Paul Lee
ECE 632 – Fall 2008
University of Virginia
<taniya, pl4u>@virginia.edu

ABSTRACT

Continual technology scaling of microprocessors has lead to emergence of hard-errors and transient faults. These hard-errors contribute towards faulty computations and shortened lifetime of microprocessors. Traditionally, architects look at microarchitecture-level to analyze the effect of hard-errors. However, by analyzing the effect of hard-errors at circuit-level granularity, we might achieve better overall reliability prediction. Considering this, we analyze a given microarchitectural structure, namely Arithmetic Logic Unit (ALU) at both microarchitecture-level and circuit-level for hard-errors to find out the gap between these two levels. Also, we propose and analyze an NBTI-aware ALU design which uses both architecture-level and circuit-level optimizations.

1. INTRODUCTION

With continued technology scaling, processors are becoming more susceptible to transient faults and hard errors. Hard errors are permanent faults that occur due to the wearing out of various hardware resources over time as a result of their usage. These failures are dependent on design-time factors such as the process parameters, wafer packaging, and processor design as well as runtime factors such as temperature and resource utilization. Some well known hard error phenomena include Electromigration (EM), Thermal Cycling (TC), Stress Migration (SM), Time Dependant Dielectric Breakdown (TDDB) and Negative Bias Temperature Instability (NBTI). Designing processors that can operate reliably in the presence of these different fault phenomena is a key challenge facing the CMOS and microprocessor industry.

To accelerate the hardware design cycle, architects often employ architectural simulators of the hardware they are designing. Similarly to quantify the effect of reliability phenomena on lifetime of the microprocessor, architects use methodology from an architectural and application perspective. For simplicity, architects often assume a microarchitectural unit as a huge block without considering circuit configuration. In this work we aim to show how circuit-level knowledge can affect architecture-level reliability simulations and solutions to those problems. To bridge this gap between architectural and circuit level reliability simulations, it is important to analyze both of them. We look into TDDB and NBTI failure techniques in an Arithmetic Logic Unit (ALU) in both levels. Also, using the PTM technology [1] we find that the effect of technology scaling is profound, especially on 22nm technology. Therefore, we need to design microarchitectural structures that would be resilient to different hard faults. Consequently, we propose an NBTI-aware ALU design at microarchitecture level as well as using some circuit-level techniques. Towards this end, this paper makes the following specific contributions:

- We analyze the architecture-level and circuit-level reliability simulation of an adder for TDDB and NBTI effects. We find out the gap between these levels and show the necessity of a combined approach to tackle reliability problems.
- We extrapolate the effect of technology scaling on TDDB and NBTI using 65nm, 45nm, 32nm and 22nm technologies.
- Finally, we propose an NBTI-aware ALU design using both microarchitectural and circuit-level techniques. We get a 60% improvement of lifetime with respect to a non-NBTI aware ALU.

Our goal in this research is to bridge the gap between microarchitecture-level and circuit-level research as these two levels are very strongly related to each other.

2. BACKGROUND

2.1 TDDB

Time-dependent dielectric breakdown (TDDB), or gate oxide breakdown, is a well studied failure mechanism in semiconductor devices. The gate dielectric wears down with time, and fails when a conductive path forms in the dielectric. The advent of thin and ultra-thin gate oxides, coupled with non-ideal scaling of supply voltage is accelerating TDDB failure rates. A recent experimental work was done at IBM [2] on TDDB where experimental data was collected over a wide range of oxide thicknesses, voltages, and temperatures to create a unified TDDB model for current and future ultra-thin gate oxides. The model shows that the lifetime due to TDDB for ultra-thin gate oxides is highly dependent on voltage and has a larger than exponential degradation due to temperature.

$$MTTF = \frac{1}{V^{(A-BT)}} e^{\frac{X+\frac{Y}{T}+ZT}{KT}} \quad \text{..... (1)}$$

Where MTTF is time to fail, A, B, X, Y and Z are constants equal to 78, -0.081, 0.759ev, -66.8evK and -8.37E-4ev/K respectively [3]. V is V_{gs} and T is temperature.

2.2 NBTI

Negative Bias Temperature Instability (NBTI) is becoming a growing concern for CMOS technologies and affects the lifetime of PMOS transistors. NBTI increases the threshold voltage (V_t) of PMOS devices, which in turn causes degradation in the speed of circuits. When a logic input of '0' is applied to the gate of a PMOS transistor ($V_{gs}=-V_{dd}$), NBTI occurs due to the generation of interface traps at the Si/SiO_2 interface. This is known as the *Stress* phase for the PMOS. The increase in V_t due to stress is given by the equation [4]:

$$\Delta V_{ts} = \left(\frac{qt_{ox}}{e_{ox}}\right)^{\frac{3}{2}} . K_1 . \sqrt{C_{ox}(V_{gs} - V_t)} . e^{\frac{-E_a}{4kT} + \frac{2(V_{GS} - V_t)}{t_{ox} E_{01}}} . T_0^{-0.25} . t_{stress}^{0.25}$$

where t_{stress} is the time under stress, t_{ox} is the oxide thickness and C_{ox} is the gate capacitance per unit area. K_1, E_a, T_0, E_{01} and k are

constants equal to 7.5 $C^{-0.5}nm^{-2.5}$, 0.49 eV, 10^{-8} s/nm^2, 0.08 V/nm and 8.6174 × 10^{-5}eV/K respectively [4].

When a logic input of '1' is applied to the gate (V_{gs} = 0), the transistor turns off eliminating some of the traps. This is known as the *Recovery* phase. The final increase of V_t after considering both the stress and recovery phases is:

$$\Delta V_t = \Delta V_{ts} \cdot \left(1 - \frac{2\xi_1 t_{0x} + \sqrt{\xi_2 e^{\frac{-E_a}{kT}} T_0 t_{rec}}}{(1+\delta)t_{ox} + \sqrt{e^{\frac{-E_a}{kT}}(t_{stress} + t_{rec})}}\right) \quad \ldots\ldots\ldots (2)$$

where t_{rec} is the time under recovery, ξ_2, ξ_1 and δ are constants equal to 0.5, 0.9 and 0.5 respectively [4].

In case of architectural simulation, the following very simplified model is being used [6]:

$$MTTF_{NBTI} = \left(\frac{1}{V_{gs}}\right)^{\gamma} \times e^{\frac{E_{anbti}}{kT}} \quad \ldots\ldots\ldots (3)$$

This simplified model is derived from the original ΔV_t equation. All the technology dependent constants are discarded and the following equation is the simplified ΔV_t equation:

$$\Delta V_t = V_{gs}^{m} \times t_{stress}^{n} \times e^{(-Ea/kT)}$$

Now NBTI lifetime is defined as the time for the variation of the threshold voltage of a device to reach a fixed value of 0.1V. Replacing this value in the above equation and after rearranging the equation, we get the equation of MTTF for NBTI (where MTTF is mean time to failure). Here γ and E_{anbti} are constants with 6 ~ 8 and 0.9 ~ 1.2 eV values respectively.

3. ARCHITECTURE -LEVEL vs. CIRCUIT-LEVEL RELIABILITY SIMULATION

3.1 EXPERIMENTAL SETUP

In this work we use 65nm process technology, supply voltage (V_{dd}) of 1.1V and $|V_t|$ of 0.2V. For architectural simulation, we simulate a 2-wide issue core which has a Reservation Update Unit (RUU) of 32 entries, a 16-entry Load-Store Queue (LSQ), 2 integer and floating-point ALUs, and 1 integer and floating-point multiplier. The memory system of this processor is composed of a 16 KB 4-way set-associative L1 D-cache and I-cache and 256 KB 4-way set-associative unified L2 cache. We use SimpleScalar 3.0 simulator [5] with Wattch [6] for modeling performance and power and HotSpot [7] for simulating the temperature behavior. We use one Integer benchmark from SPEC2000 benchmark suit [11], namely *gzip* and compile it for the Alpha ISA and use the reference input set. We perform detailed simulation of the benchmark for the first 100-million instruction SimPoints [8]. We estimate the lifetime of the first integer ALU. Depending on the area of this structure and using the model of equation (1) and (3) we project the lifetimes of the ALU due to TDDB and NBTI respectively.

For circuit-level simulation we use Cadence *spectre* simulator. As an ALU, we use Kogge-stone Adder circuit for simplicity. We sampled the average temperature of the first ALU from the architectural simulation over the entire run and use in cadence framework. We will give a brief overview of TDDB implementation in cadence followed by NBTI implementation. TDDB affects both PMOS and NMOS devices. To implement TDDB effect in cadence, due to the difficulty in simulating TDDB through time, we extract inputs from running gzip on a folder of documents using Pin[12]. After analyzing the inputs, we determine a characteristic input to simulate in Cadence. We then measure the V_{gs} of every device. As the failure of one device would cause the entire adder to incorrectly function, we calculate the MTTF based on the highest V_{gs} given different inputs. Secondly, to implement NBTI model, we use equation (2). To calculate the stress time and recovery time of the ALU, we again use the sampled utilization of the ALU from architectural simulator. We divided all the PMOS devices in multiple sets where each device in a set will suffer equal stress or recovery. We made this division by observing all the PMOS devices with multiple inputs. We can see from equation (2), change in V_t depends on previous V_t value. Consequently we need to update the value of V_t in a periodic manner where we choose this period to be 86400 seconds. We use the following schematic in our cadence framework:

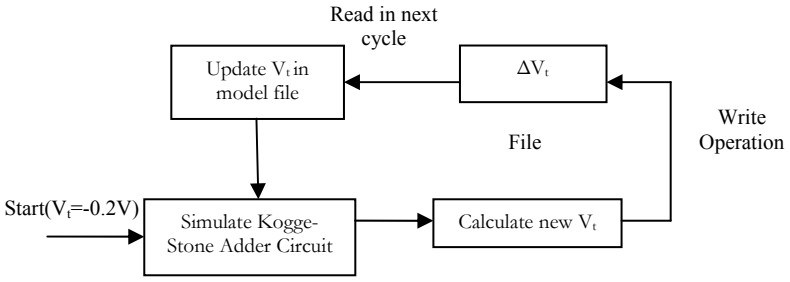

Figure 1: NBTI model in Cadence

In this experiment we start with normal threshold voltage V_t=-0.2V (for 65nm Tech), perform simulation and at each 86400 s calculate the degradation in V_t and dump it into a file. In the following cycle the data from the file is read to update V_t in PTM model file and start a new cycle simulation. To calculate a lifetime using this methodology, we assume once the circuit experiences 25% delay with respect to original output delay, the circuit is failed. Hence, we run this loop until we measure an output delay of 25%.

3.2 RESULT

TDDB: The architectural simulator gives a lifetime of 5.09 yrs. In case of cadence simulation, with multiple different inputs, we get almost 50% of all the device has V_{gs} equal to V_{dd}. As a result, we get a similar lifetime from cadence simulation too.

NBTI: The architectural simulation gives a lifetime of 5.7 yrs whereas; we get a 7 yrs lifetime from cadence simulation. From this, we can conclude architectural simulation is more conservative than cadence simulation.

To summarize, TDDB model implementation is not sensitive to architecture-level simulation or circuit-level simulation but NBTI model implementation is. The assumption made by architectural model of ΔV_t equal to 0.1 causes a circuit to fail made the model overly conservative. On top of it, this model does not account for recovery phase. A circuit has multiple PMOS devices and each of them experience both stress and recovery phase. A simplified architecture-level model is not able to capture all these phenomena without proper knowledge of circuit design. Therefore, to predict reliability properly, the assumptions should be made on the basis of circuit design. Also, knowing the circuit design might aid in proposing better solution to tackle reliability problems.

4. SCALING EFFECT

In this section we extrapolate the effect of scaling due to TDDB and NBTI on ALU. We do this using the PTM technologies in cadence framework: 65nm, 45nm, 32nm and 22nm. We use the following supply voltages for the different technologies respectively: 1.1V, 1.0V, 0.9V and 0.8V. Figure 2 and 3 present the effect of scaling for TDDB and NBTI respectively. For TDDB, we found that with decreasing technology size, the variance of V_{gs} gets larger. This is desirable since some devices with high V_{gs} are decreasing. However, a higher temperature could still cause MTTF to decrease because of the larger than exponential dependence. Also, some devices still experience maximum V_{gs} and retain the same risk of failure. For NBTI, we present the output delay in 7 yrs for each technology. The figure presents the delay for 0-to-1 transition in critical path output. The delay observed in 65nm is 25%, in 45nm it is 27%, in 32nm it is 31%, and finally 22nm shows a delay of 46%. Consequently it is very important to make NBTI-aware ALU to combat against this problem. In the following section we propose a design for NBTI-aware ALU.

	65 nm	45 nm	32 nm	22 nm
V_{gs}<Vdd/2				
Min V_{gs}	0V	0V	0V	0V
Max V_{gs}	0.255177V	0.248489V	0.24522V	0.255226V
Mean V_{gs}	0.032351V	0.033159V	0.034129V	0.035485V
StdDev V_{gs}	0.077101V	0.077045V	0.077484V	0.076488V
V_{gs}>Vdd/2				
Min V_{gs}	1.099435V	0.999175V	0.89365V	0.789839V
Max V_{gs}	1.1V	1V	0.9V	0.8V
Mean V_{gs}	1.099834V	0.99764V	0.899567V	0.797425V
StdDev V_{gs}	0.000117V	0.000181V	0.000335V	0.001789V

Figure 2: Technology Scaling Effect on TDDB

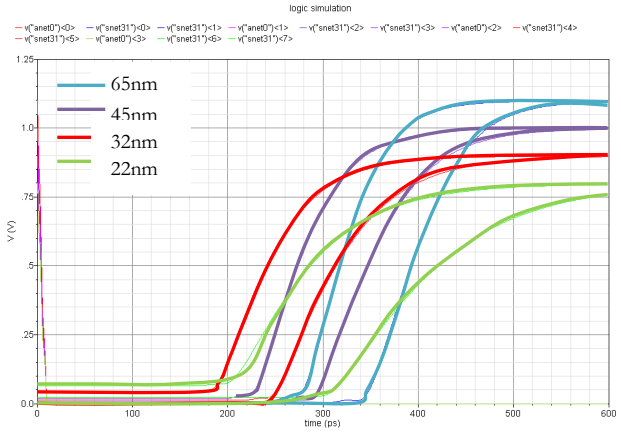

Figure 3: Technology Scaling Effect on NBTI

5. NBTI-AWARE ALU

Some recent papers have explored the effect of NBTI on a single functional unit [9, 10]. Based on the observation that functional units stay idle for a significant amount of time, these prior works propose vectoring certain 'special inputs' to these functional units which will put the functional units in recovery mode. As a particular special input will stress a particular set of PMOS, therefore multiple special inputs are vectored alternatively during the idle time. In [9], authors also proposed exploiting narrow-width operand operation. In SPEC2000 INT benchmarks, about 50% of the instructions contain operands no wider than 16 bits. They partitioned a 64-bit functional unit into four segments with granularities of 16 bits. Each segment can complete 16-bit executions independently. For normal-width values, which are wider than 16 bits, all four segments are involved in computation. In case of 16 bit operand, instruction is steered to the fastest segment. Both authors used techniques at microarchitecture level. We propose a technique that combines both microarchitecture and circuit-level optimizations.

In our design, we also utilize idle time and narrow-width operands to increase recovery time for the PMOS devices in ALU but by using *Power Gating*. *Power Gating* is a technique widely used to minimize leakage power. It is the technique wherein circuit blocks that are not in use are temporarily turned off. We apply power gating to the ALU during its idle time and also during narrow-width operation to put the PMOS devices in recovery mode. During the application of power gating, we find that the V_{gs} values of the PMOS devices stay in the range of 1.0988V~1.0995V. Therefore we can use power gating effectively to increase recovery time for the devices which also reduces leakage power and optimizes power performance. Also, in this way there is no need of vectoring special inputs which adds extra control circuits. The overhead to apply power gating would be the addition of footer circuit which is a simple NMOS device and the partitions between each block of the ALU. We partition the ALU into blocks in such a way where each block can serve narrow-width operation and rest of the blocks can stay in recovery mode. To understand the narrow-width operation frequency and to design an ALU to support this feature, we analyze the breakdown of narrow-width operation for our selected benchmark *gzip* which is presented in Table 1. By observing the table data, we divide the ALU to support 8 bits, 16 bits, 32 bits and 64 bits operands. Surprisingly, 3% of the operations have 0bit operand sizes which we use as recovery time.

Operand size	Frequency
0 bit	3%
8 bits	24%
16 bits	22%
24 bits	1%
32 bits	48%
40 bits	0%
48 bits	0%
56 bits	0%
64 bits	2%

Table 1: Narrow-width operation breakdown

As we want blocks of the ALU to experience more recovery time, we rotate the inputs among different blocks rather than using one static block for a certain bit-wide operation. Following schematic illustrates the NBTI-aware ALU.

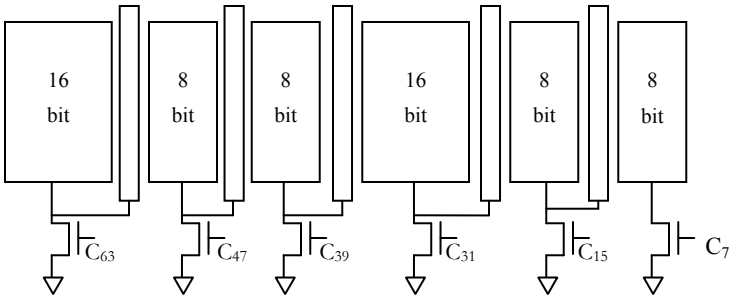

Figure 4: Schematic of a 64bit NBTI-aware ALU

From Figure 4, for 8bit operands, any one of the 8bit blocks can be used. Similarly, 16bit operations can use any one of the two 16bit blocks or can use two 8bit blocks together, 32bit operation can use either first half or the second half. Finally, 64bit operation uses the full system. C_7 to C_{63} are control signals to the footer devices to put a certain block on or off. We insert layers between blocks to safely apply power gating. These intermediate layers are made 2:1 of multiplexers. We need these multiplexers to use each block separately. The multiplexer would select input from internal circuit nodes if multiple blocks are operating together; else it would select input to operate as a single entity.

This NBTI-aware ALU gives lot of improvement over non-NBTI aware ALU. We simulated this ALU for 7 yrs and measured the critical path output delay. We get a delay of 10% only, which is actually 60% improvement over non-NBTI aware ALU. This approach not only combats NBTI effect but also reduces power consumption. The complexity of the control circuitry is less as it checks for the input operands if either all bits are '0' or all bits are '1'. Figure 5 shows the critical path 0-to-1 transition output for the NBTI-aware ALU.

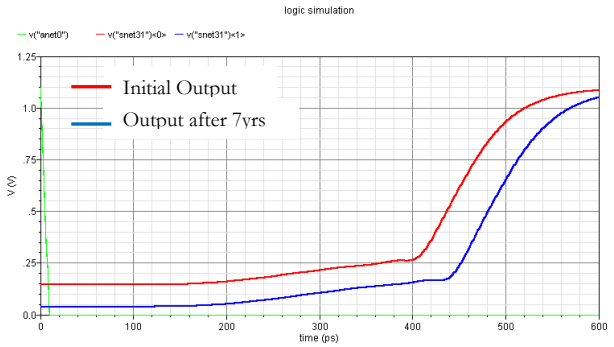

Figure 5: Critical path output of an NBTI-aware ALU

From the figure it is clear that this approach can combat NBTI problem effectively but at the cost of execution time in single cycle. As layers are inserted between blocks, it makes the adder work slower. Also, area is another factor. Therefore, this approach is a tradeoff among lifetime, power, area and execution time. If we lessen the number of intermediate layers, it would take less area, less execution time but also less lifetime and more power. Hence it is important to consider these tradeoffs and implement an efficient NBTI-aware ALU. We leave this tradeoff analysis as our future work.

6. Conclusion

Silicon reliability is one the key challenges facing the CMOS and microprocessor industry. Architects have to design processors that are resilient against lifetime reliability problems. In this paper, we explore the gap between reliability models in architecture-level and circuit-level. We show that TDDB is not sensitive to architecture or circuit level modeling but NBTI models are very sensitive. We also extrapolate the effect of scaling on NBTI and TDDB. We find 22nm technology is highly susceptible to reliability problems. Therefore, we propose an NBTI-aware ALU design which incorporates both architecture-level and circuit-level optimizations. There are tradeoffs among lifetime, power, area and execution time of the NBTI-aware ALU. As technology is scaling and becoming more susceptible to problems, it is very important to analyze the tradeoff and design an optimal ALU.

7. Acknowledgments

We would like to express our gratitude towards Dr. Ben Calhoun for stimulating discussions on NBTI-aware ALU and for answering numerous questions related to circuit-level implementation of the models. Furthermore, we would like to express our gratitude towards, Sudhanshu Khanna, for providing the design for Kogge-Stone adder.

8. References

[1] PTM web site. http://www.eas.asu.edu/ptm/.

[2] E. Y. Wu et al. Interplay of Voltage and Temperature Acceleration of Oxide Breakdown for Ultra-Thin Gate Dioxides. In *Solid-state Electronics Journal, 2002.*

[3] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors". In *Proc. of the 31st Annual International Symposium on Computer Architecture, June 2004.*

[4] W. Wang, V. Reddy and A. Krishnan, "Compact Modeling and Simulation of Circuit Reliability for 65-nm CMOS Technology", *IEEETrans. Device and Materials Reliability, 2007.*

[5] D. Burger and T. Austin. The SimpleScalar Toolset, Version 3.0. http://www.simplescalar.com.

[6] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations", *In Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 83–94, June 2000

[7] K. Skadron et al. "Temperature-Aware Microarchitecture", In *Proceedings of the International Symposium on Computer Architecture (ISCA), pages 1–13, June 2003.*

[8] T. Sherwood et al, "Automatically Characterizing Large Scale Program Behavior", In *Proceedings of the International Conference on Architectutral Support for programming languages and Operating Systems (ASPLOS), October 2002.*

[9] X. Fu, T. Li and J. Fortes, "NBTI Tolerant Microarchitecture Design in the Presence of Process Variation", In *Proc of International Symposium on Microarchitecture (MICRO), November 2008.*

[10] Jaume Abella, Xavier Vera, Antonio González, "Penelope: The NBTI-Aware Processor", In *Proc. of the 40th Annual International Symposium on Microarchitecture, Dec. 2007.*

[11] SPEC CPU2000. http://www.spec.org/cpu2000/

[12] C. Luk, et al, "Pin: building customized program analysis tools with dynamic instrumentation", in *Proc. of the 2005 ACM SIGPLAN conference on Programming language design and implementation (PLDI), June 2005.*